Section I:

AMENDMENT UNDER 37 CFR §1.121 to the CLAIMS

Claim 1 (currently amended):

A method of converting a logically ordered character stream into a character stream suitable for display by a computer and comprehension by a user, said logically ordered character stream having a plurality of characters and control codes contained within it, said method comprising:

assigning ~~in a functional programming language~~ bidirectional attributes to ~~the~~ a logical character stream;

assigning ~~in a functional programming language~~ initial level numbers and honoring any directional overrides by explicit processing;

changing attribute types based upon surrounding attribute types through weak and neutral processing ~~in a functional programming language~~;

associating final level numbers to the logical character stream through implicit processing ~~in a functional programming language~~; and

reordering said characters within said logical character stream according to said final level numbers such that said reordered characters form a character stream in display order~~, said reordering being performed in a functional programming language~~ wherein facets of layout relating to character reordering and facets related to character stream rendering are handled separately in a functional programming language, and said character stream is handled as sequential runs of integers during said steps of assigning attributes, level numbers, changing, attribute types, associating final level numbers, and reordering characters.

Claim 2 (original):

The method as set forth in Claim 1 wherein said step of assigning bidirectional attributes further comprises obtaining said bidirectional attributes from a character database.

Claim 3 (canceled).

Claim 4 (original):

The method as set forth in Claim 1 wherein said step of changing attribute types based upon surrounding attribute types through weak and neutral processing in a functional programming language comprises providing blocks of functional programming language indexed by name weak type processing, neutral type processing, and implicit level processing such that said method may be readily used as a reference.

Claim 5 (original):

The method as set forth in Claim 1 wherein one or more steps are provided in Haskell functional language.

Claim 6 (original):

The method as set forth in Claim 1 wherein one or more steps are provided in Erlang functional language.

Claim 7 (original):

The method as set forth in Claim 1 wherein one or more steps are provided in SML functional language.

Claim 8 (original):

The method as set forth in Claim 1 wherein one or more steps are provided in Miranda functional language.

Claim 9 (original):

The method as set forth in Claim 1 wherein one or more steps are provided in Lisp functional language.

Claim 10 (original):

The method as set forth in Claim 1 wherein one or more steps are provided in Scheme functional language.

Claim 11 (currently amended):

A computer readable medium encoded with software causing a computer to perform the following actions:

receiving a logically ordered character stream;

assigning ~~in a functional programming language~~ bidirectional attributes to the logical character stream;

assigning ~~in a functional programming language~~ initial level numbers and honoring any directional overrides by explicit processing;

changing attribute types based upon surrounding attribute types through weak and neutral processing ~~in a functional programming language~~;

associating final level numbers to the logical character stream through implicit processing ~~in a functional programming language~~; and

reordering said characters within said logical character stream according to said final level numbers such that said reordered characters form a character stream in display order, ~~said reordering being performed in a functional programming language~~ wherein facets of layout relating to character reordering and facets related to character stream rendering are handled separately in a functional programming language, and said character stream is handled as sequential runs of integers during said steps of assigning attributes, level numbers, changing, attribute types, associating final level numbers, and reordering characters.

Claim 12 (original):

The computer readable medium as set forth in Claim 11 wherein said software for performing said assignment of bidirectional attributes further comprises software for obtaining said bidirectional attributes from a character database.

Claim 13 (canceled):

Claim 14 (original):

The computer readable medium as set forth in Claim 11 wherein said software for performing the action of changing attribute types based upon surrounding attribute types through weak and neutral processing in a functional programming language comprises software organized into blocks of functional programming language indexed by name weak type processing, neutral type processing, and implicit level processing such that said method may be readily used as a reference.

Claim 15 (original):

The computer readable medium as set forth in Claim 11 wherein said software is Haskell functional language.

Claim 16 (original):

The computer readable medium as set forth in Claim 11 wherein said software is Erlang functional language.

Claim 17 (original):

The computer readable medium as set forth in Claim 11 wherein said software is SML functional language.

Claim 18 (original):

The computer readable medium as set forth in Claim 11 wherein said software is Miranda functional language.

Claim 19 (original):

The computer readable medium as set forth in Claim 11 wherein said software is Lisp functional language.

Claim 20 (original):

The computer readable medium as set forth in Claim 11 wherein said software is Scheme functional language.

Claim 21 (currently amended):

A text code conversion system for converting logically ordered text streams and displaying said text streams in a display order, said system comprising:

a character stream receiver for receiving a logically ordered character stream;

a bidirectional attribute assignor realized ~~in a functional programming language~~ for assigning bidirectional attributes to a received logical character stream;

an initial level assignor ~~realized in a functional programming language~~ for assigning initial level numbers and for honoring any directional overrides by explicit processing;

an attribute type changer realized ~~in a functional programming language~~ for changing attribute types based upon surrounding attribute types through weak and neutral;

a final level assignor realized ~~in a functional programming language~~ for associating final level numbers to the logical character stream through implicit processing; and

a character resequencer realized ~~in a functional programming language~~ for reordering said characters within said logical character stream according to said final level numbers such that said reordered characters form a character stream in display order, wherein facets of layout relating to character reordering and facets related to character stream rendering are handled separately, and said logically ordered character stream is handled as sequential runs of integers by said character stream receiver, said attribute assignor, said initial level assoignor, said type changer, said final level assignor, and said character resequencer, each of which are realized in a functional programming language.

Claim 22 (original):

The text code conversion system as set forth in Claim 21 wherein said bidirectional attributes assignor is adapted to obtain said bidirectional attributes from a character database.

Claim 23 (canceled).

Claim 24 (original):

The text code conversion system as set forth in Claim 21 wherein said attribute type changer attribute type changer comprises blocks of functional programming language indexed by name weak type processing, neutral type processing, and implicit level processing such that said method may be readily used as a reference.

Claim 25 (original):

The text code conversion system as set forth in Claim 21 wherein said bidirectional attribute assignor, initial level assignor, attribute type changer, final level assignor, and character resequencer comprise Haskell functional language.

Claim 26 (original):

The text code conversion system as set forth in Claim 21 wherein said bidirectional attribute assignor, initial level assignor, attribute type changer, final level assignor, and character resequencer comprise Erlang functional language.

Claim 27 (original):

The text code conversion system as set forth in Claim 21 wherein said bidirectional attribute assignor, initial level assignor, attribute type changer, final level assignor, and character resequencer comprise SML functional language.

Claim 28 (original):

The text code conversion system as set forth in Claim 21 wherein said bidirectional attribute assignor, initial level assignor, attribute type changer, final level assignor, and character resequencer comprise Miranda functional language.

Claim 29 (original):

The text code conversion system as set forth in Claim 21 wherein said bidirectional attribute assignor, initial level assignor, attribute type changer, final level assignor, and character resequencer comprise Lisp functional language.

Claim 30 (original):

The text code conversion system as set forth in Claim 21 wherein said bidirectional attribute assignor, initial level assignor, attribute type changer, final level assignor, and character resequencer comprise Scheme functional language.